Práctica de Diseño de Software

Grado en Ingeniería Informática
Mención Ingeniería del Software
Universidad de Valladolid

curso 2018-2019

Capítulo 1

Supuesto práctico

Para consultar las normas de entrega vaya al final del documento (página 20).

1.1. Adaptado del enunciado de la asignatura Modelado de Software en el curso 2018-2019

En este proyecto se persigue realizar el diseño e implementación (parcial) correspondiente a la automatización de una empresa de Repostería. Dicha empresa tiene una tienda (simplificación respecto del enunciado de Modelado) y un horno para realizar lo que denominamos "productos de horno".

La empresa realiza compras para el funcionamiento de sus tareas. En esta simplificación solamente se ha incluido la compra de materias primas necesarias para la elaboración de "productos de horno". De la misma forma, se entiende que la empresa puede vender en su tienda diferentes productos, algunos productos elaborados (no necesariamente de horno), velas de cumpleaños, etc. y en particular los llamados "productos de horno" que son los únicos que se incluyen en esta simplificación.

Los proveedores de la empresa emiten facturas que son pagadas por el Encargado cada cierto tiempo. En este enunciado no se indica el máximo de tiempo establecido por la empresa para realizar el pago a proveedores, con el objetivo de evitar restricciones.

Los clientes pueden realizar pedidos de productos de horno, indicando la fecha para la que lo necesitan. En el momento en el que un cliente hace un pedido, éste se encuentra en estado "registrado". Los empleados que trabajan en el horno consultan pedidos de los clientes con las fechas en las que se necesitan y priorizan la elaboración. Cuando un empleado de horno se pone a elaborar el pedido, modifica el estado del mismo que pasa ser "preparando". Cuando lo da por terminado, modifica el estado para pasar a ser "terminado". Cuando se completa la elaboración de un producto de horno, según sean los pasos en la receta se decrementan las existencias de la cantidad de materia prima necesaria para su elaboración.

Los empleados de horno solamente modifican el estado del pedido de "registrado" a "preparando", y de "preparando" a "terminado". Cuando el cliente viene a la tienda a recoger su pedido, el dependiente modifica su estado a "recogido" y lo cobra mediante la realización de una venta. Dicha venta puede contener otros productos además del pago del pedido.

1.1.1. Restricciones adicionales

El sistema deberá utilizar una base de datos, cuyo diseño se aporta, implementada con Derby.

Deberá asegurarse que todos los usuarios están previamente identificados en el sistema para acceder a cualquier función, y que las funciones serán las que correspondan al usuario según su rol en la empresa.

...

Se ha decidido realizar una aplicación de escritorio con acceso a BD. En teoría la BD sería centralizada en modo cliente-servidor, pero para facilitar las prácticas en este caso supondremos la conexión a una BD local. Todo de ello de forma que sea inmediato realizar los cambios necesarios para que la BD pase a ser remota y centralizada.

Se han realizado las fases de Elicitación y Especificación de Requisitos, se han descrito los requisitos funcionales, no funcionales. De la fase de análisis se ha realizado el Modelado del Dominio y la especificación de casos de uso en análisis. A continuación se aporta una documentación parcial de análisis que incluye: el Modelo del Dominio, un fragmento del diagrama de casos de uso y la especificación de cuatro de los casos de uso. Se aporta también el diseño de la base de datos (diseño lógico y físico) realizado en la primera fase del Diseño a partir del modelado conceptual expresado en el Modelo de Dominio.

Capítulo 2

Documentación parcial de Análisis

2.1. Vista parcial del diagrama de Casos de Uso

Se aporta una vista parcial del diagrama de Casos de Uso. Para esta práctica se han escogido aquellos que se encuentran resaltados y para los que se adjunta una especificación.

Se han definido distintos actores. Los que vamos a considerar en este trabajo son:

- el actor generalización Empleado
- el actor Dependiente
- el actor Empleado de Horno
- el actor Encargado

En la Figura 2.1 se muestra el mencionado fragmento del diagrama de casos de uso. Tal como se aprecia en dicha figura, para este trabajo vamos a tener en cuenta los siguientes casos de uso:

Empleado: Identificarse

Dependiente Registrar venta directa

Empleado de Horno Preparar Pedido de Horno

Encargado Consultar facturas pendientes de pago

Aclaraciones sobre el caso de uso "Identificarse": El empleado se identifica con su dni y contraseña. Si todo va bien, se carga su perfil con las opciones que puede realizar este tipo de empleado. Para esta práctica, a cada tipo de actor se le mostrará una lista de opciones. Esta lista debe contener al menos el CU que se pide realizar para ese actor en este supuesto práctico. El resto de opciones estarán identificadas con el nombre del actor y un número. Su funcionalidad asociada será mostrar un mensaje que informe de "Opción aún no implementada".

Los casos de uso no sombreados no se realizarán aunque tenga una relación de include/extend con los casos de usos principales.

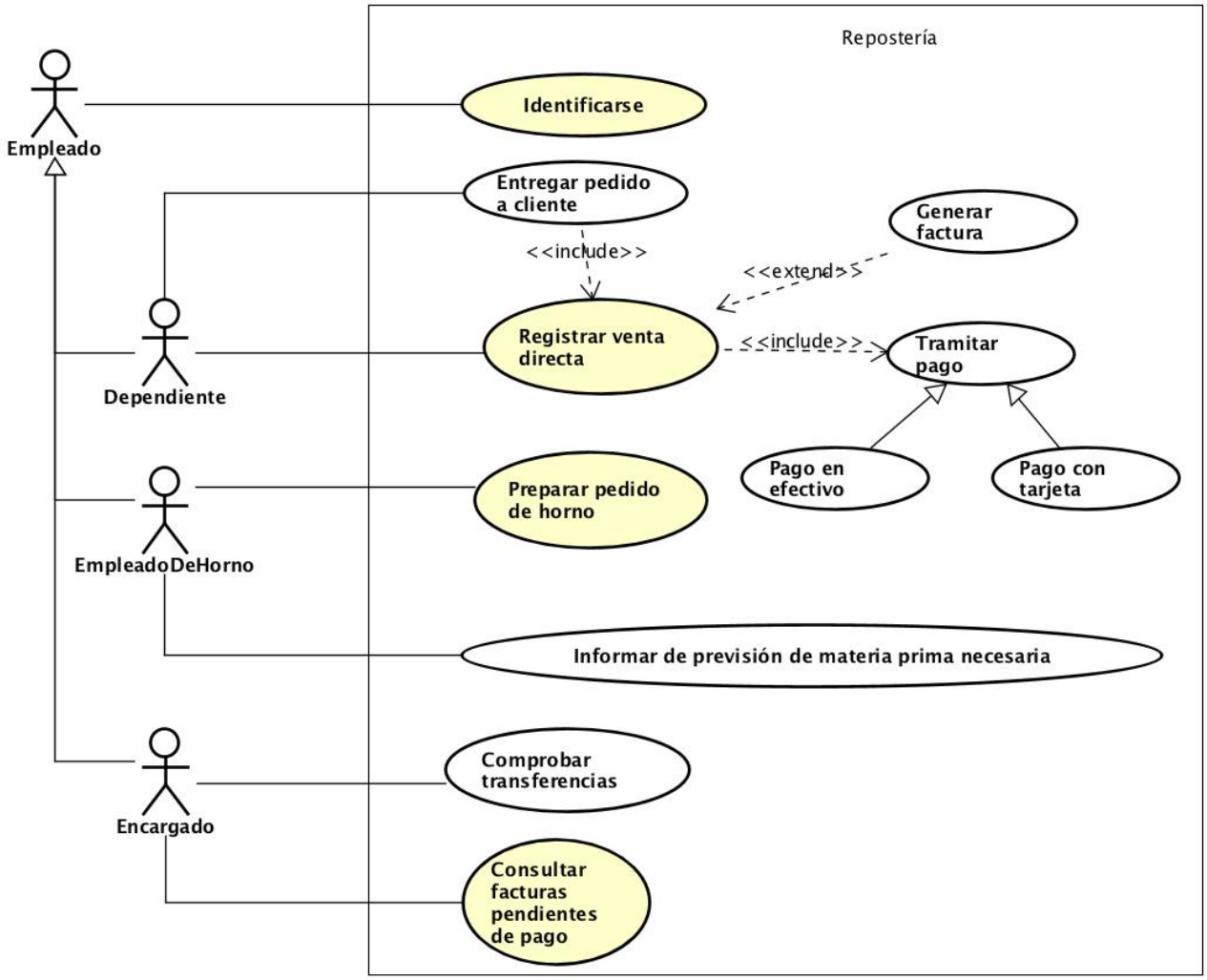

Figura 2.1: Fragmento del Diagrama de casos de uso. Se muestran sólo algunos ejemplos por actor

2.2. Modelo del dominio

En la fase de análisis se ha obtenido el modelo del dominio que se muestra en la Figura 2.2. En dicho diagrama se ha omitido a propósito la especificación de restricciones OCL que sí era objetivo de la asignatura de Modelado de Sistemas Software. Debe tenerse en cuenta que éste es una simplificación a partir del supuesto práctico de la asignatura de Modelado. Nótese que no se han incorporado en este modelo de dominio algunos atributos de información y clases. El objetivo es simplificar. Por otra parte es necesario aclarar que el modelo que aquí se presenta no puede ser considerado como solución a un supuesto evaluable en la asignatura de Modelado.

En los siguientes apartados se describen los escenarios de los casos de uso mencionados.

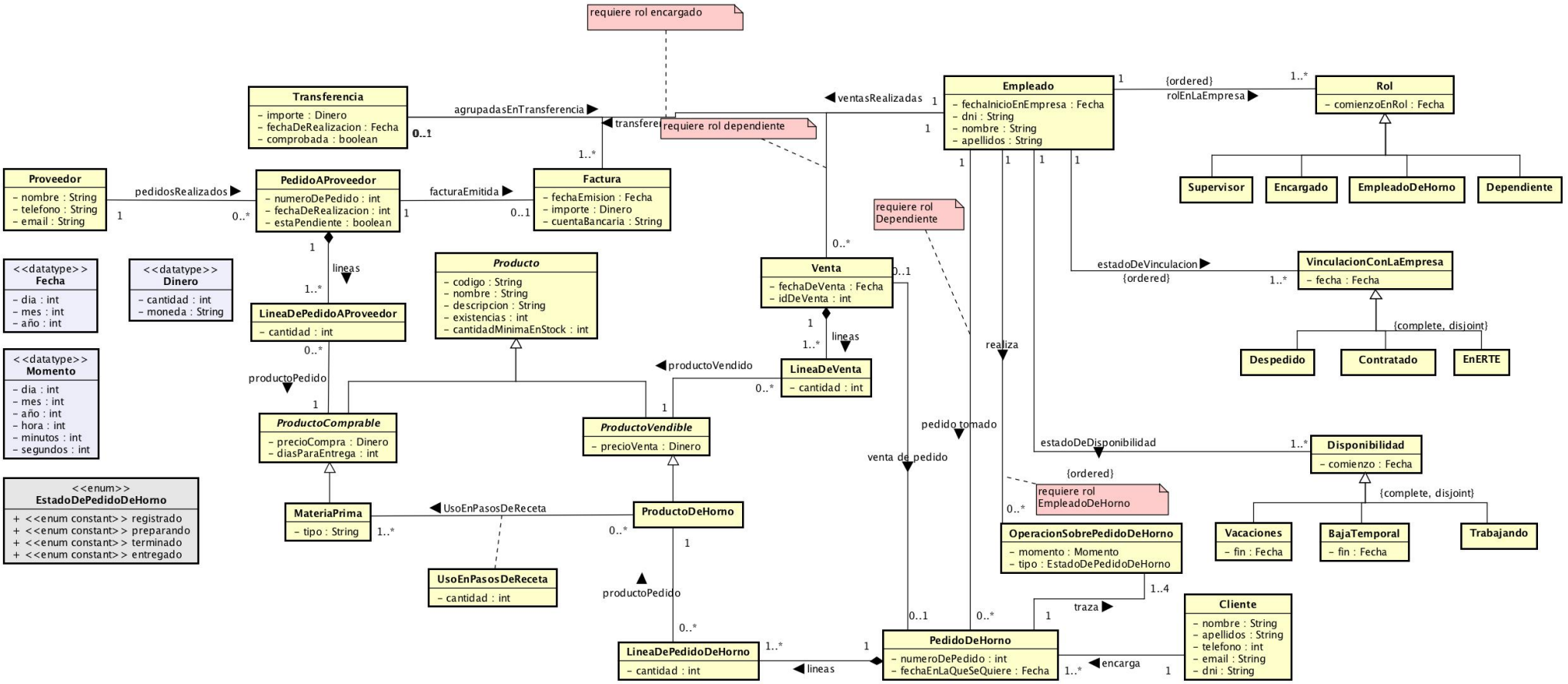

Figura 2.2: Modelo del Dominio. Punto de partida del diagrama de clases inicial

2.3. Especificación de casos de uso

De la misma forma que se aclaró en el apartado relativo al Modelo del Dominio, la especificación de casos de uso que aquí se describe no se corresponde exactamente con lo esperado en el supuesto planteado en la asignatura de Modelado. Se han realizado muchas simplificaciones para que no sea necesario trabajar con demasiadas clases de dominio ni tablas en la base de datos.

2.3.1. Empleado: Identificarse

Actor: Empleado

Caso de Uso: Identificarse

Precondición: El actor no se encuentra identificado en el sistema

Secuencia normal:

1. El actor Empleado introduce dni y contraseña.
2. El sistema comprueba que dicho par dni-contraseña corresponden con un empleado en activo y muestra las opciones correspondientes a su rol en el sistema.

Alternativas y excepciones:

(2 a) Si no existe empleado con dni indicado, el sistema muestra un mensaje de error, a continuación el caso de uso queda sin efecto.

(2 b) Si la contraseña es incorrecta para el empleado con el dni indicado, el sistema muestra un mensaje de error, a continuación el caso de uso queda sin efecto.

(2 c) Si el empleado no está en activo (activo es: contratado y trabajando), el sistema muestra un mensaje de error y a continuación el caso de uso queda sin efecto.

Postcondición: El empleado está identificado en el sistema y sus opciones de trabajo mostradas.

2.3.2. Encargado: Consultar facturas pendientes de pago

Actor: Encargado

Caso de Uso: Consultar facturas pendientes de pago

Precondición: El actor Encargado se encuentra identificado en el sistema

Secuencia normal:

1. El actor Encargado introduce un rango de fechas, fecha inicial y fecha final, para las que desea obtener las facturas que aún están pendientes de pago. Estas fechas se refieren la fecha de emisión de la factura.
2. El sistema comprueba que se trata de un rango de fechas válido.
3. El actor Encargado introduce un proveedor para consultar las facturas pendientes de pago a dicho proveedor.
4. El sistema comprueba que se trata de un proveedor válido.
5. El sistema muestra un informe con la información de las facturas en el rango de fechas dado, del proveedor indicado. En particular muestra, para cada resultado, el nombre del proveedor, el número de pedido, el importe de la factura y las fechas de realización del pedido y de emisión de la factura, respectivamente.

Alternativas y excepciones:

(1a), (3a) El actor cancela y el caso de uso queda sin efecto.

(1b) El actor indica que desea información sobre todas las facturas pendientes de pago del año actual y el caso de uso continúa por el paso 3.

(1c) El actor indica que desea información sobre todas las facturas pendientes de pago en general, independientemente de la fecha de emisión, y el caso de uso continúa por el paso 3.

(2a) El sistema comprueba que se trata de una fecha inicial incorrecta, informa al usuario del error y el caso de uso vuelve al paso 1.

(2b) El sistema comprueba que se trata de una fecha final incorrecta, informa al usuario del error y el caso de uso vuelve al paso 1.

(2c) El sistema comprueba que la fecha final no es posterior o igual a la inicial, informa al usuario del error y el caso de uso vuelve al paso 1.

(3b) El actor indica que desea información sobre las facturas de todos los proveedores y el caso de uso continúa por el paso 5.

(4a) El sistema comprueba que no tiene registrado ningún proveedor como el indicado, informa al usuario del error y el caso de uso vuelve al paso 3.

(5a) El sistema comprueba que no hay facturas pendientes de pago que coincidan con los datos de la consulta, informa al usuario que no se ha encontrado información y el caso de uso finaliza.

Postcondición: Se ha producido un informe para el usuario pero no se ha modificado el estado del sistema.

Indicaciones de usabilidad: el actor podrá elegir que los resultados se muestren ordenados según la fecha o según el proveedor (cada criterio podrá ser elegido en primer o segundo lugar).

2.3.3. Empleado de horno: Preparar pedido de horno

Actor: Empleado de Horno

Caso de Uso: Preparar pedido de horno

Precondición: El actor Empleado de Horno se encuentra identificado en el sistema.

Secuencia normal:

1. El actor solicita la lista de pedidos de horno que deben estar para hoy o mañana.
2. El sistema muestra todos los pedidos de horno en estado "registrado" y con fecha para la que se desea el pedido en el día de hoy o en el de mañana.
3. El actor selecciona un pedido.
4. El sistema muestra el detalle del pedido: cada producto de horno en el pedido con su cantidad y pide confirmación de que se va a preparar dicho pedido.
5. El actor indica que se comienza a preparar el pedido.
6. El sistema registra la operación "preparando" sobre el pedido en el momento actual y el caso de uso finaliza.

Alternativas y excepciones:

(1a), (3a), (5a) El actor cancela y el caso de uso queda sin efecto.

(2a) El sistema comprueba que no hay pedidos de horno pendientes de preparar para hoy o para mañana, muestra un mensaje y el caso de uso queda sin efecto.

(5b) El actor indica "Otro pedido" y el caso de uso vuelve al paso 2.

(6a) El sistema comprueba que no hay existencias de Materia Prima suficientes para realizar el pedido, muestra un mensaje informativo y el caso de uso vuelve al paso 2.

Postcondición: Se ha registrado la operación "preparando" para el pedido seleccionado.

Indicaciones de usabilidad: Se recomienda mostrar al usuario los pedidos ordenador por fecha y cantidad de elementos en el pedido en el paso 2. Se recomienda mostrar el detalle de las materias primas que faltan en el mensaje informativo de (6a)

2.3.4. Dependiente: Registrar venta directa

Actor: Dependiente

Caso de Uso: Registrar venta directa

Precondición: El actor Dependiente se encuentra identificado en el sistema

Secuencia normal:

1. El actor Dependiente introduce el código de un producto y la cantidad.
2. El sistema comprueba que se trata de un código de producto correcto y una cantidad correcta, registra la línea de venta y muestra nombre del producto, precio de venta al público y subtotal (precio por cantidad).
3. El actor Dependiente indica que ha finalizado la venta.
4. El sistema presenta el total a pagar.
5. **Se realiza el CU "Tramitar Pago"**.
6. El actor Dependiente indica que no es necesaria factura.
7. El sistema registra la venta completa, actualiza las existencias y el caso de uso finaliza.

Alternativas y excepciones:

(1a), (3a), (6a) El actor cancela y el caso de uso queda sin efecto.

(2a) El sistema comprueba que no se tiene un producto en venta con el código introducido, informa el error al usuario y el caso de uso continua por el paso 1.

(2b) El sistema comprueba que no hay existencias del producto indicado, informa el error al usuario y el caso de uso continua por el paso 1.

(2c) El sistema comprueba que la cantidad introducida es un número entero mayor que 0, informa el error al usuario y el caso de uso continua por el paso 1.

(2d) El sistema comprueba que la cantidad introducida es mayor que las existencias del producto (teniendo en cuenta las posibles líneas de venta anteriores), informa el error al usuario y el caso de uso continua por el paso 1.

(3a) El actor Dependiente indica que la venta no ha finalizado y el caso de uso continua por el paso 1.

(6a) Si el actor Dependiente indica que el cliente necesita factura **Se realiza el CU "Generar Factura"** y a continuación el caso de uso continua por el paso 7.

Postcondición: Se ha registrado una nueva venta y al menos una línea de venta integrada en ella. Se han actualizado las existencias de los productos vendidos en cada línea de venta.

Indicaciones de usabilidad: Para simplificar no se tienen en cuenta modificaciones durante la venta, retirar una línea de venta, modificar una cantidad, etc. Tampoco se tiene en cuenta que se puede leer el código del producto mediante dispositivos lectores de código de barras, etc.

2.4. Restricciones a tener en cuenta en la implementación

El sistema deberá utilizar una base de datos cuyo esquema (diseño lógico) se aporta en la página 14, implementada con Derby. Deberá utilizarse el script de creación de la base de datos (diseño físico) se aporta en la página 15. El sistema será una aplicación de escritorio independiente del sistema operativo que se desarrollará en JAVA, jdk 8 utilizando Swing y JDBC. El sistema será desarrollado en NETBEANS en la misma versión que se encuentra instalada actualmente en los laboratorios de la Escuela (actualmente Netbeans 8.2).

Capítulo 3

Diseño de Datos

3.1. Diseño lógico (datamodel style)

En la Figura 3.1 se muestra el diseño de los datos como esquema relacional (diseño lógico), utilizando una representación basada en UML con estereotipos. Los estereotipos «PK» y «FK» representan claves primarias y claves foráneas, respectivamente. Los clasificadores con el estereotipo «table» representan una tabla en el esquema relacional. Las tablas que representan un elemento del dominio se identifican con el estereotipo «entity». Las tablas que representa asociaciones se identifican con el estereotipo «association». Las tablas que representan valores enumerados se identifican con el estereotipo «enum» y aparecen sombreadas. La transformación al modelo relacional de la relación de herencia (Producto) presente en el Modelo del Dominio se ha realizado mediante aplanamiento hacia arriba, creando una única tabla Producto. El modelado de roles temporales presentes en el Modelo del Dominio para Empleado se ha realizado mediante tablas asociación y tablas enum.

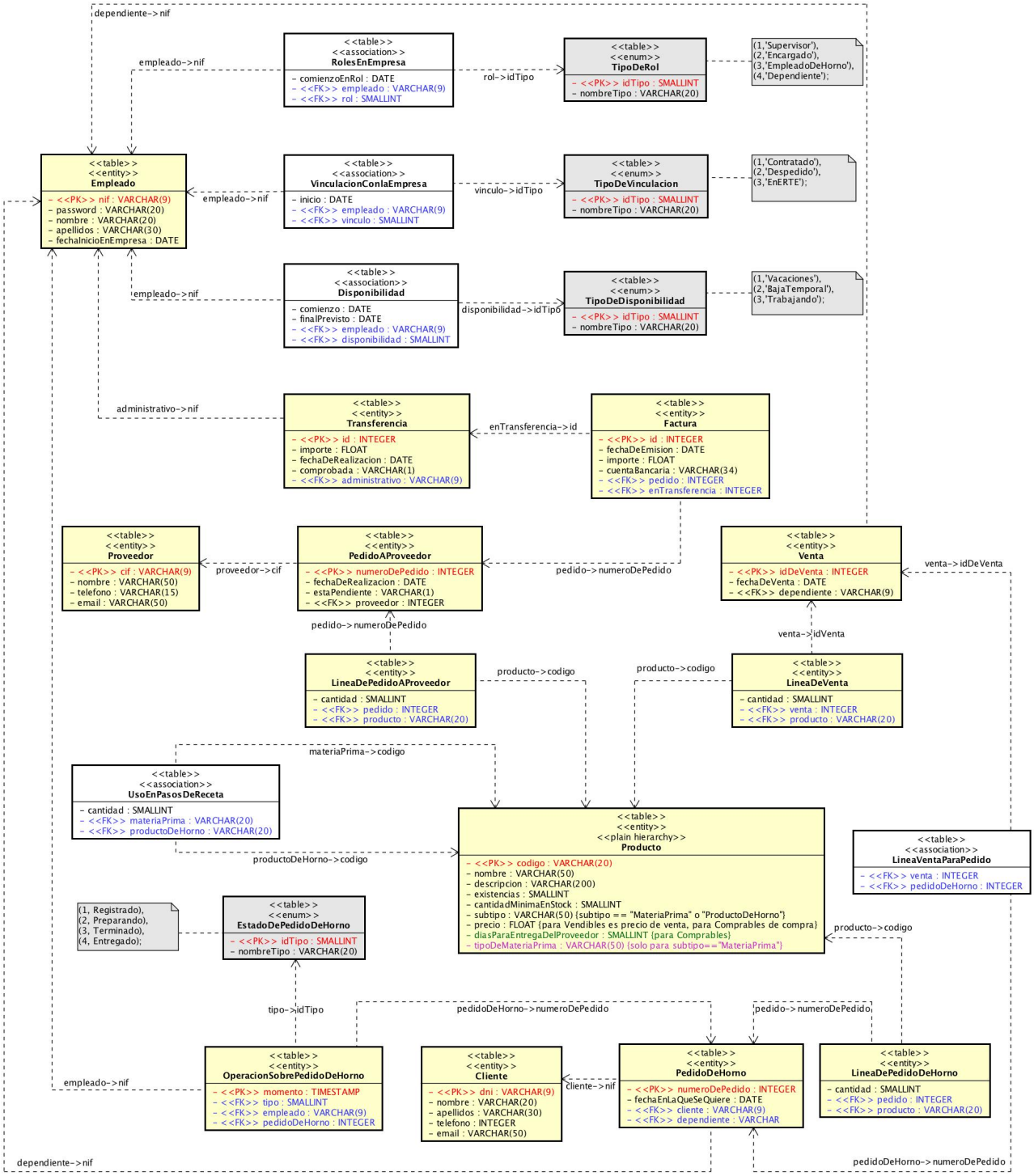

Figura 3.1: Diseño lógico de la base de datos relacional.

3.2. Scripts de creación de la base de datos

Este script puede obtenerse en un archivo almacenado como recurso en el aula virtual siguiendo el enlace: `https://aulas.inf.uva.es/mod/resource/view.php?id=25830`.

```
--Derby does not support DROP TABLE IF EXISTS VENTA

DROP TABLE OPERACIONSOBREPEDIDODEHORNO;
DROP TABLE ESTADODEPEDIDODEHORNO;
DROP TABLE LINEAVENTAPARAPEDIDO;
DROP TABLE LINEADEPEDIDODEHORNO;
DROP TABLE PEDIDODEHORNO;
DROP TABLE CLIENTE;

DROP TABLE LINEADEVENTA;
DROP TABLE VENTA;

DROP TABLE FACTURA;
DROP TABLE TRANSFERENCIA;

DROP TABLE LINEADEPEDIDOAPROVEEDOR;
DROP TABLE PEDIDOAPROVEEDOR;
DROP TABLE PROVEEDOR;

DROP TABLE USOENPASOSDERECETA;
DROP TABLE PRODUCTO;

DROP TABLE DISPONIBILIDADEMPLEADO;
DROP TABLE VINCULACIONCONLAEMPRESA;
DROP TABLE ROLESENEMPRESA;
DROP TABLE EMPLEADO;

DROP TABLE TIPODEDISPONIBILIDAD;
DROP TABLE TIPODEVINCULACION;
DROP TABLE TIPODEROL;

-- Enum
create table TIPODEROL
(
        IdTipo SMALLINT not null,
        nombreTipo VARCHAR(20) not null,
                PRIMARY KEY(IdTipo)
);

INSERT INTO TIPODEROL
VALUES  (1,'Supervisor'),
        (2,'Administrativo'),
        (3,'Operario'),
        (4,'Dependiente');

-- Enum
create table TIPODEVINCULACION
(
        IdTipo SMALLINT not null,
        NombreTipo VARCHAR(20) not null unique,
                PRIMARY KEY(IdTipo)

);

INSERT INTO TIPODEVINCULACION
VALUES  (1,'Contratado'),
        (2,'Despedido'),
        (3,'EnERTE');

-- Enum
```

```sql
create table TIPODEDISPONIBILIDAD
(
        IdTipo SMALLINT not null,
        NombreTipo VARCHAR(20) not null unique,
                PRIMARY KEY(IdTipo)
);

INSERT INTO TIPODEDISPONIBILIDAD
VALUES  (1,'Vacaciones'),
        (2,'BajaTemporal'),
        (3, 'Trabajando');

-- Entity
create table EMPLEADO
(
        Nif VARCHAR(9) not null,
        Password VARCHAR(20) not null,
        Nombre VARCHAR(20) not null,
        Apellidos VARCHAR(30) not null,
        FechaInicioEnEmpresa DATE not null,
                PRIMARY KEY(Nif)
);

-- Association
create table ROLESENEMPRESA
(
        ComienzoEnRol DATE not null,
        Empleado VARCHAR(9) not null,
        Rol SMALLINT not null,
            FOREIGN KEY(Empleado) REFERENCES EMPLEADO(Nif),
            FOREIGN KEY(Rol) REFERENCES TIPODEROL(IdTipo)
);

-- Association
create table VINCULACIONCONLAEMPRESA
(
        inicio DATE not null,
        Empleado VARCHAR(9) not null,
        Vinculo SMALLINT not null,
                FOREIGN KEY(Empleado) REFERENCES EMPLEADO(Nif),
                FOREIGN KEY(Vinculo) REFERENCES TIPODEVINCULACION(IdTipo)
);

-- Association
create table DISPONIBILIDADEMPLEADO
(
        Comienzo DATE not null,
        FinalPrevisto DATE,
        Empleado VARCHAR(9) not null,
        Disponibilidad SMALLINT not null,
                FOREIGN KEY(Empleado) REFERENCES EMPLEADO(Nif),
                FOREIGN KEY(Disponibilidad) REFERENCES TIPODEDISPONIBILIDAD(IdTipo)
);

-- Entity
create table PRODUCTO
(
        Codigo VARCHAR(20) not null,
        Nombre VARCHAR(50) not null,
        Descripcion VARCHAR(200) not null,
        Existencias SMALLINT not null,
        CantidadMinimaEnStock SMALLINT not null,
        Subtipo VARCHAR(50) not null, -- {subtipo=="MateriaPrima" o subtipo=="
            ProductoDeHorno"}
        Precio FLOAT not null, -- {para Vendibles es precio de venta, para
            Comprables precio de compra}
```

```
        DiasParaEntregaDelProveedor SMALLINT, -- {not null para Comprables, null
            para Vendibles}
        TipoDeMateriaPrima VARCHAR(50), --{not null para subtipo=="MateriaPrima",
            null para subtipo!="MateriaPrima"}
                PRIMARY KEY(Codigo)
);

-- Association
create table USOENPASOSDERECETA
(
        Cantidad SMALLINT,
        MateriaPrima VARCHAR(20),
        ProductoDeHorno VARCHAR(20),
                FOREIGN KEY MateriaPrima REFERENCES PRODUCTO(Codigo),
                FOREIGN KEY ProductoDeHorno REFERENCES PRODUCTO(Codigo)
);

-- Entity
create table PROVEEDOR
(
        Cif VARCHAR(9) not null,
        Nombre VARCHAR(50) not null,
        Telefono VARCHAR(15) not null,
        Email VARCHAR(50) not null,
                PRIMARY KEY(Cif)
);

-- Entity
create table PEDIDOAPROVEEDOR
(
        NumeroDePedido INTEGER not null,
        FechaDeRealizacion DATE not null,
        EstaPendiente VARCHAR(1) not null,
        Proveedor VARCHAR(9) not null,
                PRIMARY KEY(NumeroDePedido),
                FOREIGN KEY(Proveedor) REFERENCES PROVEEDOR(Cif)
);

-- Entity
create table LINEADEPEDIDOAPROVEEDOR
(
        Cantidad SMALLINT not null,
        Pedido INTEGER not null,
        Producto VARCHAR(20) not null,
                FOREIGN KEY(Pedido) REFERENCES PEDIDOAPROVEEDOR(NumeroDePedido),
                FOREIGN KEY(Producto) REFERENCES PRODUCTO(Codigo)
);

-- Entity
create table TRANSFERENCIA
(
        Id INTEGER not null,
        Importe FLOAT not null,
        FechaDeRealizacion DATE not null,
        Comprobada VARCHAR(1) not null,
        Administrativo VARCHAR(9) not null,
                PRIMARY KEY(Id),
                FOREIGN KEY(Administrativo) REFERENCES EMPLEADO(Nif)
);

-- Entity
create table FACTURA
(
        Id INTEGER not null,
        FechaDeEmision DATE not null,
        Importe FLOAT not null,
```

```
        CuentaBancaria VARCHAR(34) not null,
        Pedido INTEGER not null,
        EnTransferencia INTEGER, --null si no se ha realizado transferencia
                PRIMARY KEY(Id),
                FOREIGN KEY(EnTransferencia) REFERENCES TRANSFERENCIA(Id),
                FOREIGN KEY(Pedido) REFERENCES PEDIDOAPROVEEDOR(NumeroDePedido)
);

-- Entity
create table VENTA
(
        IdDeVenta INTEGER not null,
        FechaDeVenta DATE not null,
        Dependiente VARCHAR(9) not null,
                PRIMARY KEY(IdDeVenta),
                FOREIGN KEY(Dependiente) REFERENCES EMPLEADO(Nif)

);

-- Entity
create table LINEADEVENTA
(
        Cantidad SMALLINT not null,
        Venta INTEGER not null,
        Producto VARCHAR(20) not null,
                FOREIGN KEY(Venta) REFERENCES VENTA(IdDeVenta),
                FOREIGN KEY(Producto) REFERENCES PRODUCTO(Codigo)
);

-- Entity
create table CLIENTE
(
        Nif VARCHAR(9) not null,
        Nombre VARCHAR(20) not null,
        Apellidos VARCHAR(30) not null,
        Telefono INTEGER,
        EMail VARCHAR(50),
                PRIMARY KEY(Nif)
);

-- Entity
create table PEDIDODEHORNO
(
        NumeroDePedido INTEGER not null,
        FechaEnLaQueSeQuiere DATE not null,
        Cliente VARCHAR(9) not null,
        Dependiente VARCHAR(9) not null,
                PRIMARY KEY(NumeroDePedido),
                FOREIGN KEY(Cliente) REFERENCES CLIENTE(Nif),
                FOREIGN KEY(Dependiente) REFERENCES EMPLEADO(Nif)
);

-- Entity
create table LINEADEPEDIDODEHORNO
(
        Cantidad SMALLINT not null,
        Pedido INTEGER not null,
        Producto VARCHAR(20) not null,
                FOREIGN KEY(Pedido) REFERENCES PEDIDODEHORNO(NumeroDePedido),
                FOREIGN KEY(Producto) REFERENCES PRODUCTO(Codigo)
);

-- Association
create table LINEAVENTAPARAPEDIDO
(
        Venta INTEGER not null,
```

```
        Pedido INTEGER not null,
                FOREIGN KEY(Venta) REFERENCES VENTA(IdDeVenta),
                FOREIGN KEY(Pedido) REFERENCES PEDIDODEHORNO(NumeroDePedido)

);

-- Enum
create table ESTADODEPEDIDODEHORNO
(
        IdTipo SMALLINT not null,
        nombreTipo VARCHAR(20) not null,
                PRIMARY KEY(IdTipo)
);

INSERT INTO ESTADODEPEDIDODEHORNO
VALUES  (1,'Registrado'),
        (2,'Preparando'),
        (3,'Terminado'),
        (4,'Entregado');

-- Entity
create table OPERACIONSOBREPEDIDODEHORNO
(
    Momento TIMESTAMP not null,
    Tipo SMALLINT not null,
    Empleado VARCHAR(9) not null,
    PedidoDeHorno INTEGER not null,
        PRIMARY KEY(Momento),
        FOREIGN KEY(Tipo) REFERENCES ESTADODEPEDIDODEHORNO(IdTipo),
        FOREIGN KEY(Empleado) REFERENCES EMPLEADO(Nif),
        FOREIGN KEY(PedidoDeHorno) REFERENCES PEDIDODEHORNO(NumeroDePedido)
);
```

Capítulo 4

Normas de entrega y criterios de evaluación

4.1. Indicaciones para la entrega

El seguimiento del proyecto se realizará a través de la aplicación Pivotal Tracker, `www.pivotaltracker.com`. Todos los alumnos se habrán creado una cuenta en Pivotal Tracker y habrán sido añadidos al proyecto correspondiente según su equipo de trabajo. El alumno deberá estar identificado con su correo en alumnos.uva.es y su identificación como login en los laboratorios de la Escuela.

Todos los equipos disponen de un canal de comunicación en la instancia rocket de la Escuela (`https://rocket.inf.uva.es`).

Para realizar la entrega se preparará una *release* en Pivotal Tracker. En dicha *release* se adjuntará un archivo zip o tgz (o tar.gz) con el contenido que se especifica en el apartado 4.2.

Se deja a decisión de los alumnos miembros de un equipo, que la entrega correspondiente a la *release* se realice mediante un repositorio en el GitLab de la Escuela (cumpliendo estrictamente la estructura y contenidos que se especifican en el apartado 4.2). Esto último es opcional ya que no forma parte de los contenidos de esta asignatura pero puede ser preferencia del alumnado que ya domina estas tecnologías trabajar de esta forma. Para esto último deberán seguirse las siguientes normas:

- La entrega se realizará añadiendo a la profesora (usuario `yania`) con permisos de tipo `Reporter` al repositorio que contiene el proyecto en el GitLab de la Escuela cuando ya no se vaya a realizar ningún *commit+push*. Cualquier *push* al repositorio una vez vencido el plazo de entrega será penalizado con 0 en la Práctica en la convocatoria correspondiente.
- El enlace (url) al repositorio y cualquier documentación necesaria al respecto serán anotadas en la *release* en Pivotal Tracker.

4.2. Estructura y contenidos de la entrega

La entrega tendrá la siguiente estructura de carpetas:

```
models/
app/
```

En la carpeta `models` se encontrará un archivo Astah Professional o Visual Paradigm (a elección del equipo). Los modelos y diagramas contenidos en dicho archivo Astah Professional o Visual Paradigm que son evaluables corresponden a lo realizado en la fase de diseño. Se tendrá cuidado de alojar todo lo realizado en un modelo llamado Diseño organizado con sus respectivos submodelos y diagramas.

Se contará con los diagramas necesarios para representar la arquitectura de referencia, los estilos universales decomposition style, uses style, inheritance style y data model así como los diagramas de clases de diseño detallado y los diagramas de secuencia con la realización en diseño de los casos de

uso que se piden en este enunciado. Adicionalmente, se aportará un diagrama de estados para modelar la interfaz de usuario del sistema (en el que se modelará únicamente lo necesario para los casos de uso que se especifican en este documento). Los diagramas tendrán que ser legibles y comprensibles. Si los diagramas se hacen excesivamente grandes deberán utilizarse los elementos que ofrece UML para reducir el tamaño y la complejidad de los modelos.

En la carpeta `app` se espera una estructura como la siguiente:

```
app/netbeansProject/
app/db/
```

La carpeta `app/netbeansProject` contendrá, como su nombre indica, el proyecto NETBEANS que implementa el caso de uso. La implementación no debe incumplir el diseño propuesto.

La carpeta `app/db` contendrá un archivo config.db que indicará la url con el puerto, el nombre de la base de datos, el usuario y el password. Es decir, lo necesario para conectarse con la BD. En dicha carpeta también se encontrarán unos scripts que permitirán la regeneración de la base de datos (el suministrado con este enunciado) así como tantos scripts SQL como sean necesarios para poblar automáticamente la base de datos de cara a probar la aplicación (puede ser uno o varios separados por casos de uso o escenarios) siempre que se documente apropiadamente el propósito de cada uno.

Ni en el control de versiones, ni en los archivos de entrega debe residir la base de datos Derby.

Fecha límite para la entrega

A continuación se especifican las fechas límite de entrega para cada convocatoria. Esta será la fecha de la *release* en PIVOTAL TRACKER. Si no se cumple la fecha límite, el equipo será penalizado con 0 en la Práctica en la convocatoria correspondiente.

convocatoria ordinaria 29 de mayo de 2019

convocatoria extraordinaria 19 de junio de 2019

4.3. Criterios de Evaluación

Se valorará:

(a) la aplicación y consistencia en el diseño de la arquitectura de 3 capas (capas estrictas) combinada con MVC, se puede considerar además añadir una capa transversal de servicios (capa relajada);

(b) la aplicación de los patrones GRASP y algunos patrones de diseño conocidos, en particular la aplicación de patrones de acceso a datos;

(c) la corrección y completitud de los modelos UML.

(d) la calidad de la solución.

Los criterios anteriores tendrán el mayor peso en la evaluación de la práctica (70 %) y se desglosan de la siguiente forma:

- diagramas de la arquitectura de referencia y descomposición modular- 0,5/10
- diagramas de dependencias entre capas - 0,4/10
  - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Identificarse? 0,1/10
  - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Registrar venta directa? - 0,1/10
  - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Preparar Pedido de Horno? - 0,1/10

    - ¿Se diseñan adecuadamente las dependencias necesarias para el CU Consultar facturas pendientes de pago? - 0,1/10

- diagramas de clases de diseño detallado - 0,5/10
- diagrama de estados que modela la interfaz - 0,1/10
- Diagramas de Secuencia Realización en Diseño-
    - ¿Se diseña adecuadamente la realización del CU Identificarse (incluye mecanismo de persistencia)? - 1/10
    - ¿Se diseña adecuadamente la realización del CU Registrar venta directa (incluye mecanismo de persistencia)?- 1,5/10
    - ¿Se diseña adecuadamente la realización del CU Preparar pedido de horno (incluye mecanismo de persistencia)? - 1,5/10
    - ¿Se diseña adecuadamente la realización del CU Consultar facturas pendientes de pago)? - 1,5/10

Respecto del código de la aplicación (30 %) se valorará:

(e) que sea consistente con el diseño arquitectónico (1/10);

(f) que sea consistente con el modelo dinámico diseñado (es decir, con los diagramas de secuencia que describen la realización en diseño de los casos de uso) (1/10);

(g) que se comporte según lo esperado en las situaciones válidas y que no acepte situaciones inválidas comunicando al usuario los errores que se controlan (pruebas de aceptación superadas) (1/10).

La calidad y la seguridad del código de la aplicación será analizada mediante SonarQube. Los problemas de calidad y seguridad detectados por esta herramienta conducirán a una penalización de hasta un 10 % en la nota correspondiente al código de la aplicación (que es el 30 % del total).

La forma de uso de Pivotal Tracker no será evaluable, pero el uso será obligatorio. En caso de no utilizarse, la práctica no podrá ser considerada entregada.

El uso de rocket no será evaluable ni obligatorio, aunque es recomendable para introducir debates en los que pueda participar el profesor y dar sugerencias o donde preguntar dudas que pueda resolver el profesor.